Demosaicing as an ill-posed inverse problem

Study programme: Mathematics
Study branch: General Mathematics

Prague 2020
I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ............ date ............ ........................................

Author’s signature
I would like to thank my supervisor, Filip Šroubek, for his guidance, patience, and resources. I would also like to thank Keigo Hirakawa for kindly providing the source code to his demosaicing method. A warm thanks also belongs to my closest friends and family for their support and calming presence.
Title: Demosaicing as an ill-posed inverse problem

Author: Veronika Mariničová

Department: Department of Software and Computer Science Education

Supervisor: doc. Ing. Filip Šroubek, Ph.D., DSc., Department of Software and Computer Science Education

Abstract: Color information of a scene is only recorded partially by a digital camera. Specifically, only one of the red, green, and blue color components is sampled at each pixel. The missing color values must be estimated – a process called demosaicing. Demosaicing can be solved as an individual step in the image processing pipeline. In this case, any errors and artefacts produced by this step are carried over into further steps in the image processing pipeline and are possibly magnified. Alternatively, we can try to resolve several degradations at once in a joint solution, which eliminates this effect. We present one such solution, that in addition to demosaicing, also jointly solves denoising, deconvolution, and super-resolution in the form of a convex optimization problem. We provide an overview of demosaicing methods and evaluate the results from our solution against selected existing methods.

Keywords: Bayer pattern, color filter array, inverse problem, demosaicing, super-resolution, deconvolution
# Contents

**Introduction**  
2

1 Demosaicing Methods  
1.1 Notation 5  
1.2 Linear Interpolation Methods 7  
1.2.1 Non-adaptive Methods 7  
1.2.2 Adaptive methods 8  
1.3 Mathematical Optimization Methods 13  
1.3.1 Projection onto Convex Sets 13  
1.3.2 Inverse Problem Approaches 14

2 Joint Demosaicing, Denoising, Deconvolution, and Super-resolution  
2.1 Formulation of the Problem 16  
2.2 Convex Optimization 16  
2.3 Alternating Direction Method of Multipliers 17  
2.3.1 Convergence 18  
2.4 Application of ADMM 19

3 Experimental Results  
3.1 Simulation of the Raw Output 21  
3.2 Quantitative Comparison 21  
3.3 Visual Comparison 22  
3.4 Summary 25

Conclusion  30

Bibliography  31

List of Figures  34

List of Tables  35
Introduction

To fully reconstruct a color image with a digital camera, we need at least three color components for each pixel. This can be achieved in two ways.

One option is to have three sensors, CMOS or CCD, in a camera and use a beam splitter to project light onto each sensor and subsequently record values for all three color components for each pixel. Such cameras are costly, however, due to the sensors themselves and the precision required for their perfect alignment. Therefore, this type of digital camera is less common [1, 2].

The other, much more common option, is to have a single sensor overlaid with a color filter array (CFA). This way, we record the value of only one color component for each pixel location in the resulting image. Which color component is recorded is determined by the pattern (mosaic) of the overlaid CFA. The most commonly used CFA is the Bayer CFA, which can be seen in Figure 1. Here, as well as in most CFA patterns, the green filter takes up twice as many pixel locations as either of the other two color filters. This is because the human eye is more sensitive to mid-range color frequencies, corresponding to the green part of the spectrum [2]. Other types of CFAs are also known. Some use the CMY color model with cyan, magenta and yellow filters. These CMY filters behave better in low-light conditions, as the light only has to pass through one layer of filters, whereas the traditional RGB filters are usually created by combining CMY filters [3]. Other CFAs add a panchromatic filter to their pattern, which lets through the whole visible spectrum. The sensor under a panchromatic filter then measures luminance (perceived brightness), which helps reduce noise in the reconstructed image, especially in low-light conditions [4].

For single-sensor cameras, each sensor position only measures the value of one color component. To reconstruct the full color information – a process called demosaicing – we need to estimate the remaining two color components at each pixel. The simplest and probably most obvious way to do this is by estimating the values using linear interpolation in each of the three color layers independently. This method, however, is far from optimal. It increases the occurrence of visible artefacts, for example the zipperizing effect, which appears when we incorrectly interpolate across edges, rather than along them [5]. This artefact can be seen in Figure 2b, where the original scene in Figure 2a – consisting of a sharp blue-yellow

![Figure 1: Bayer Color Filter Array](image-url)

2
color edge – is reconstructed by bilinear interpolation in each of the three color planes. Another prominent artefact is the false color artefact, which can be seen in Figure [3b] where the snow-covered roof has visible artificial color patterns. This artefact is caused by neglecting correlation between the color components [5].

To achieve a more visually pleasing result, a more specialized approach is needed. In practice, a better result can be achieved by consciously choosing a direction in which we interpolate. We should take edges and different texture features in each color plane into account and interpolate within them, so as not to disrupt them. Inter-channel correlation must be taken into account as well. For example if there is an edge in the red channel, there is a high chance of there being an edge in the two other channels as well, and any misalignment can cause the appearance of false colors. We describe several of these specific approaches to demosaicing in Chapter [4].

Demosaicing is not the only problem in image processing. When a digital image is formed, the original scene undergoes significant degradation. The degradation can be caused by several factors. These factors include blur, which is introduced to the scene when the light intensity is integrated on the sensor (sensor blur), as well as by any camera motion (motion blur). Then there is the effect of sampling itself. If sampling density is too low for a given scene, the resulting image can lack a significant amount of detail. Finally, there is also random additive noise. Therefore the image processing pipeline often includes, besides demosaicing, several other reconstruction steps: deconvolution to remove the sensor or motion blur, denoising to reduce noise, super-resolution, if a higher resolution image is needed, and demosaicing to obtain full color information.

In this thesis, we will review existing methods for demosaicing in Chapter [1]. Then in Chapter [2] we will formulate an optimization problem, that performs demosaicing, deconvolution, denoising and super-resolution jointly. In the same chapter, we will also describe a method that is used to solve convex optimization problems. Finally, we will compare the presented approach against some of the methods described in Chapter [1] on images from the Kodak image dataset [6] and use the PSNR and SSIM metrics – described in Chapter [3] – to assess the quality of the results.
Figure 2: Crisp original image in (a) and visible zippering effect on bilinearly interpolated image in (b).

Figure 3: Original scene in (a) and visible false color artefact on reconstructed image in (b).
1. Demosaicing Methods

Numerous different approaches towards demosaicing have been described [7, 8, 9, 10, 11, 12]. While the initial effort was mainly to improve linear interpolation methods in order to reduce the visible artefacts while maintaining the advantage of quick computation, new and better methods were quickly introduced. These methods include the use of discrete derivatives of the color channels, defined later in Section 1.2.2 to decide which direction to interpolate in; as well as some basic smoothing strategies like median filtering, which very successfully reduces the zippering artefact. Other methods include formulating demosaicing as an optimization problem [11, 12, 13, 14]. There are also frequency domain approaches [14] and machine learning models [15].

For CFA-based, single-sensor cameras, the problem of demosaicing can also be seen as super-resolution – the problem concerned with increasing the resolution of an image. Because the CFA subsamples the original image, as seen in Figure 1.2, the obtained color channels are only available in a reduced resolution. By applying a super-resolution method on each of the color channels, we obtain their full-resolution versions and therefore also a full-resolution, full-color reconstructed image. Traditionally, super-resolution is done when we have several images of the same scene available, all slightly shifted from one another [16]. The shift can be due to camera shake, movement of the object, and so on. The specific movement is then estimated and the images must be transformed back to align on top of each other. Having the low-resolution images aligned allows us to interpolate the pixels onto a higher-resolution grid.

In the case of demosaicing, however, we only have information about the scene from a single image and need to fill in the gaps introduced by the CFA. Furthermore, this approach does not consider dependencies between the respective color channels. We will show later in this chapter that considering the color channels as independent is a suboptimal approach.

It is possible to divide the different demosaicing methods into two groups. The first consists of linear interpolation based methods, and the second consists of methods that reconstruct the original scene by solving a mathematical optimization problem [1].

1.1 Notation

The camera takes the original scene represented by the color picture in Figure 1.1a and samples the color channels using the Bayer CFA, as shown in Figure 1.2. For our purposes, we will assume the raw output of a digital camera to be an RGB image as shown in Figure 1.1b. Notice the overall green coloring of the raw output image. This is due to the fact that the green component is sampled at a higher rate. On an enlarged cutout from the raw output in Figure 1.1c we can clearly see the mosaic effect of the Bayer CFA. Later in this chapter we will denote the raw output by \( O \), and the original scene by \( F \).

We will refer to the red, green and blue color channels of the full reconstructed image as \( R \), \( G \), \( B \), respectively. Consider a pixel in the raw output image, the value of which corresponds to the measured green (red, blue) component. We will
Figure 1.1: Original scene (a), mosaiced raw output from a single-sensor digital camera (b), and zoomed-in raw output.

Figure 1.2: Subsampled color channels obtained from Bayer’s CFA.
refer to such a pixel as a green (red, blue) pixel. We will denote them by \( r_{i,j}, g_{i,j}, \) and \( b_{i,j}, \) where the subscripts are the spatial coordinates of the pixel in the raw output image as well as in the final reconstructed image. We will differentiate two types of rows in the image. A red row will be a row where there are green and red pixels, a blue row will be a row where there are green and blue pixels. For any given pixel, we must now estimate the value of the two components that were not recorded. We will denote any estimated values by \( \hat{r}_{i,j}, \hat{g}_{i,j}, \) and \( \hat{b}_{i,j}. \) We will denote an estimate of the full image by \( \hat{F}. \)

Furthermore, as in most of the publications cited in this thesis, see for example [17], we will refer to the green component of the image as the luminance channel. In the RGB color space, luminance is defined as a linear combination of the three color components, where the green component holds the majority of the weight [12]. Luminance represents the brightness, while chrominance represents the color. We will sometimes refer to the red and blue components as the chrominance channels or components.

### 1.2 Linear Interpolation Methods

In this section, we will take a closer look at some straightforward methods which for the most part only try to improve linear interpolation. Some general assumptions about color images are explored, for example that the hue is slowly varying throughout an image [2]. These simple methods can be further divided into non-adaptive and adaptive methods [2].

#### 1.2.1 Non-adaptive Methods

Non-adaptive methods do not adjust the interpolation scheme depending on local texture in the scene. For the purpose of a comfortable introduction to the subject, we will represent this group by a single method: bilinear interpolation. More sophisticated approaches exist and so this method is rarely ever used in its purest form. Nevertheless, thanks to its straightforwardness and low computational cost, it can be considered the founding stone for other methods.
Bilinear Interpolation

Let us consider a red pixel \( r_{i,j} \), see Figure 1.3 for reference. The goal is to obtain \( \hat{b}_{i,j} \) and \( \hat{g}_{i,j} \) – the blue and green components. We estimate the green component by bilinear interpolation as [18]:

\[
\hat{g}_{i,j} = \frac{1}{4}(g_{i-1,j} + g_{i+1,j} + g_{i,j-1} + g_{i,j+1}).
\]

We get the estimate for the blue component in a similar fashion, only we average the diagonal neighbors instead of the horizontal and vertical neighbors, see again Figure 1.3. Values on borders of the image are estimated only using the neighboring pixels that fall within the image boundaries. As could perhaps be expected, the results are less than ideal. The main cause of the bad quality is that the local texture of the scene is not taken into account – edges and other texture features are disregarded [5]. While this approach may perform sufficiently for grayscale images and smooth parts of color images, the resulting color images suffer from severe color artefacts [2].

An improvement on the above described naive approach was introduced by Freeman [19]. Let us consider the channels \( R - G \) and \( B - G \) which are channels obtained by subtracting the luminance component \( G \) from the chrominance components. Areas suffering from the zippering artefact will show a spike or an irregularity in these color difference channels. This can be seen in Figure 1.4, where the \( B - G \) channel of the original scene is smooth, while the \( B - G \) channel of the reconstructed image shows bright stripes that precisely copy the shape of the false color effect, that can be seen in Figure 1.4b. Note that we have increased the brightness for better visibility. These planes are therefore smoothed by simple median filtering, which removes, or at least smooths out, these spikes. This smoothed difference channel is then added back to the interpolated luminance channel, which gives us a smooth red (blue) component. Subsequent composition of these new smoothed components yields us an image with reduced occurrence of color artefacts [17]. The new reconstructed image can be seen in Figure 1.5b. While the artefacts have not disappeared completely, the overall result is much better. This useful trick can be effective in reducing color artefacts introduced by any demosaicing method [5].

1.2.2 Adaptive methods

Constant-hue Interpolation

A group of early demosaicing methods [8, 20] was based on a simple assumption that images with smooth texture have a constant hue. In this thesis, when we talk about hue, we will understand it as a property of color which lets us differentiate between different color shades. The assumption of constant hue falls apart in regions of complex texture, or near edges. However within object boundaries, it works well. We will represent the concept of hue as the ratios of colors: \( \frac{R}{G} \) and \( \frac{B}{G} \). We want to keep these ratios constant [2] [17], and so for two pixel locations \( [i, j] \) and \( [k, l] \) within a small neighborhood, we will assume [8]:

\[
\frac{r_{i,j}}{g_{i,j}} = \frac{r_{k,l}}{g_{k,l}}.
\]
Figure 1.4: Original scene (a), corrupted reconstructed image (b), $B - G$ channels of the original scene (c), and of the reconstructed image (d).
In order to work with hue, we must have a full luminance channel, and so the first step is to fill in missing values in the luminance channel. This can be done through bilinear interpolation. We then perform interpolation in the color ratio channel and get the missing chrominance component as shown for an example pixel location below:

\[
\hat{r}_{2,3} = g_{2,3} \cdot \frac{1}{2} \left( \frac{r_{1,3}}{g_{1,3}} + \frac{r_{3,3}}{g_{3,3}} \right).
\]

Similarly, the hue of horizontal neighbors is averaged for estimating the red component at a green pixel in a red row, and the hue of four diagonal neighbors is averaged to estimate the red component at a blue pixel. The blue component is estimated in an analogous fashion.

**Homogeneity-directed interpolation**

A similar approach was chosen in a newer method [9], where we first construct horizontally and vertically interpolated images \(F_H, F_V\). For each pixel neighborhood \(B_{i,j}(\delta)\) of predefined pixel size \(\delta\), we take pixels whose color is no further than \(\epsilon\) in the CIELAB color space. We denote the set of such pixels by \(U_{i,j}(\delta, \epsilon)\). The homogeneity of a pixel is then given as follows:

\[
H_{i,j}(\delta, \epsilon) = \frac{|U_{i,j}(\delta, \epsilon)|}{|B_{i,j}(\delta)|},
\]

where \(|U_{i,j}(\delta, \epsilon)|\) and \(|B_{i,j}(\delta)|\) denote the number of pixels in \(B_{i,j}(\delta)\) and \(U_{i,j}(\delta, \epsilon)\), respectively. The CIELAB color space is best suited for this appli-
cation, as the distance between two colors best matches the visual difference perceived by people [2].

Suppose we have a pixel in an area suffering from a color artefact, as can be seen in Figure [21]. Such an area will not have a constant hue. On the contrary, the colors will be quickly varying. Thus the homogeneity value of such a pixel would be small. We want to avoid this scenario and so we will always try to maximize the homogeneity [9, 21]. The idea now is to choose for each pixel the corresponding pixel value from either $F_H$ or $F_V$ by comparing their homogeneity values and choosing in favor of the greater one. The last step is to use median filtering as a smoothing technique as described in Freeman’s method [19].

**Edge-directed interpolation**

Significant improvement over the methods above is achieved with some consideration. If we could decide for every pixel whether there is an edge going through it and in which direction, we could avoid interpolating across it and therefore preserve the local texture of the scene. In fact, we can do that very easily, using discrete partial derivatives. Let us first define them.

In digital image processing, we often define partial first order derivatives of an image in a single given component at a location $f_{x,y}$ as follows [22]:

$$\frac{\partial f}{\partial x} = f(x + 1, y) - f(x - 1, y)$$

$$\frac{\partial f}{\partial y} = f(x, y + 1) - f(x, y - 1)$$

Similarly, we define second order partial derivatives as:

$$\frac{\partial^2 f}{\partial x^2} = f(x - 1, y) - 2f(x, y) + f(x + 1, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y - 1) - 2f(x, y) + f(x, y + 1)$$

Prescott and Laroche’s proposed a three step method [23], that relies on discrete derivatives. The second order discrete derivatives – in absolute value and halved in magnitude – are used as edge detectors that help us in choosing the correct direction for interpolation in the luminance channel $G$. The horizontal and vertical edge detectors $h$ and $v$ look as follows:

$$h = \left| \frac{r_{i,j-2} + r_{i,j+2}}{2} - r_{i,j} \right| \quad (1.1)$$

$$v = \left| \frac{r_{i-2,j} + r_{i+2,j}}{2} - r_{i,j} \right| \quad (1.2)$$

The edge indicators are defined in a similar fashion for a blue pixel location. We compare the two edge detectors and interpolate in the direction of the smaller one – along an edge, not across it. If the values of the edge detectors are equal, we interpolate bilinearly in both directions.
Therefore the estimated green component for a specific example location $\hat{g}_{3,3}$ is given by the following formula:

$$\hat{g}_{3,3} = \begin{cases} 
\frac{g_2 + g_4}{2}, & \text{if } h(3, 3) < v(3, 3) \\
\frac{g_2 + g_3 + g_4}{4}, & \text{if } h(3, 3) > v(3, 3) \\
\frac{g_2 + g_3 + g_4}{4}, & \text{if } h(3, 3) = v(3, 3)
\end{cases}$$  \hspace{1cm} (1.3)

It is possible in Equation 1.3 to compare the derivatives against a chosen threshold instead against each other to make for a stronger condition, as was presented by Hibbard [24]. In the same work, Hibbard uses derivatives computed from the green neighboring pixels to determine the direction of interpolation in the chrominance channels. We now have a full luminance channel. In the next step, color difference channels are interpolated: $R - G$ and $B - G$, as in Section 1.2.2. We estimate the red chrominance channel using $R - G$ channel for different types of pixels: green pixel in a red row, green pixel in a blue row, and a blue pixel, respectively, as follows:

$$\hat{r}_{3,2} = \frac{(r_{3,1} - \hat{g}_{3,1}) + (r_{3,3} - \hat{g}_{3,3})}{2} + g_{3,2},$$

$$\hat{r}_{2,3} = \frac{(r_{1,3} - \hat{g}_{1,3}) + (r_{3,3} - \hat{g}_{3,3})}{2} + g_{2,3},$$

$$\hat{r}_{2,2} = \frac{(r_{1,1} - \hat{g}_{1,1}) + (r_{1,3} - \hat{g}_{1,3}) + (r_{3,1} - \hat{g}_{3,1}) + (r_{3,3} - \hat{g}_{3,3})}{4} + \hat{g}_{2,2}.$$

Missing blue components are estimated similarly.

This approach was expanded upon by Hamilton and Adams [7], where instead of using second order partial derivatives as in Equations 1.1 - 1.2 as a classifier for an edge or increased texture, we use a sum of three terms. The first are second order partial derivatives in the chrominance channel, red or blue, depending on which pixel type we are estimating the green component at. The second term are first order partial derivatives in the luminance channel, and finally the last term is a color difference correction term. These classifiers are then first compared to a relatively low threshold to determine the overall smoothness of the area, and then to a higher threshold to assess whether there is a rapid detail change in the scene. The green value at a blue/red pixel is estimated similarly as above, by an average of neighboring green pixels. In this case, however, a properly scaled second order derivative of the color channel is added. Similarly, a scalar of second order derivative of the luminance channel is added to the average of chrominance components when estimating the red and blue channels.

**Other adaptive methods**

Many adaptive methods are based on the ones described in Section 1.2.2. Often they will use an edge directed approach to reconstruct the luminance channel and then describe a new approach for reconstructing the chrominance information [21]. It is also not rare for these methods to combine any of the approaches described above, for example Kimmel [20] combined edge indicators with constant color ratios.
1.3 Mathematical Optimization Methods

In the second group of demosaicing methods we take our \textit{a priori} assumptions about the scene and together with our observed data, we use them to find the optimal solution of a defined mathematical problem \cite{21}.

1.3.1 Projection onto Convex Sets

In 2002, Gunturk et al. \cite{10} introduced an iterative method, and made two important observations. The first one is, that the color channels in images are highly correlated – the highest correlation being in high frequencies. For simplicity, we will intuitively understand high frequencies in images to be edges and textured areas – areas where the scene changes rapidly, therefore with a \textit{high frequency}. We can then interpret this observation to mean that edges and textures are mostly located in the same place across all color channels.

The second observation is that because it is sampled at a higher rate, the green channel is much less aliased\footnote{An effect introduced by undersampling a function, corrupting it and adding false information \cite{22}} than the two chrominance channels. We later turn this into a constraint to enforce the chrominance channels to have high frequency information whenever the luminance channel does.

We start with an initial estimate – which can be given by bilinear or edge directed interpolation. At each iteration in the algorithm, the estimate is projected onto two constraint sets. The first projection is very straightforward. We want to use the observed data, so whenever we have a value in any channel obtained from the camera sensor, we force it into the estimate. The second projection constraint was essentially described above. We decompose the estimate into frequency subbands that contain either high or low frequencies. We then force the high frequency components in the chrominance channels R and B to be similar to the high frequency component of the luminance channel G.

Since the two constraint sets are shown to be convex, the method converges to a feasible solution in their intersection.

A similar thought was explored in Li’s method in 2005 \cite{25}, where the color difference planes $R - G$ and $B - G$ are interpolated iteratively, exploiting the fact that each update in either $G$ or a $R - G$ ($B - G$) can be used to improve the estimates of the other planes from the previous iteration. Li proposed that the zippering effect can occur when the algorithm is stopped after too few iterations in a high frequency area, or when the algorithm is stopped too late – and thus over enforcing the color ratio rule – for smooth areas that are near edges. Therefore he used a discrete Laplacian operator $L$:

$$L = \frac{1}{4} \cdot \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix},$$

which for any given pixel gives us the average of the second order partial derivatives in the horizontal and vertical direction. Second order derivatives are more sensitive to fine detail, than first order derivatives. Therefore this operator is useful in emphasizing textured and detailed high frequency areas \cite{22}. The Laplacian
operator, as defined in Equation 1.4, is applied to color difference channels $R - G$ and $B - G$ which gives us texture channels $T_R$, $T_B$. A pixel location $[i, j]$ is then pronounced a high frequency location, if $T_R[i, j]$ or $T_B[i, j]$ are higher than a given threshold. Otherwise it is said to be a low frequency component. An adaptive stopping criterion was then introduced. The computation at a given location terminates when the error between $(n+1)$th and $n$th iteration falls below a threshold which is set high for high frequency locations and low for low frequency ones.

1.3.2 Inverse Problem Approaches

Let us denote the observed data obtained from the camera sensor by $O$ as a sample of the original full scene denoted by $F$, which is corrupted by additive noise $N$. We can then formulate the relationship between the observed data and the original image as follows [26]:

$$O = SF + N,$$

where $S$ is the linear operator that performs subsampling according to the CFA. We can therefore interpret demosaicing as the inverse problem to Equation 1.5. Examples of such approaches include [11, 12, 13, 14].

Regularization

Regularization is a way to turn an ill-posed mathematical problem into a well-posed one by exploiting a priori knowledge about the image. This is not a new concept in image processing. Regularization is often used for problems like denoising, deconvolution and super-resolution. It is therefore not surprising that some of the methods we will describe combine some of these problems and solve them jointly [26]. The generic formulation of such an optimization problem can be formulated as follows:

$$F^* = \arg \min_{\hat{F}} \gamma d(O, \hat{F}) + \sum_i \mu_i c_i(\hat{F}),$$

where $d(O, \hat{F})$, called the data-fidelity term, captures the magnitude of error between the observed data and our estimate and $c_i$ are constraints through which we enforce our a priori knowledge. The best estimate and the result of the reconstruction is denoted by $F^*$. The weight of the constraints as well as the data fidelity is controlled by the parameters $\gamma$ and $\mu_i$. The set of constraints can incorporate things like overall smoothness of the image, as well as correlation of the color channels, as is presented in [11]. There, Keren and Osadchy find the solution by minimizing the cost function defined below:

$$\text{COST} = \text{ROUGH} + \lambda (\text{CHROMA TERM}),$$

where ROUGH is the term enforcing smoothness and consists of the sum of second order discrete derivatives, and CHROMA TERM is the term controlling, through a positive parameter $\lambda$, the correlation of color channels. Here it is represented as the sum of squared norms of all vector products of $(r_{i,j}, g_{i,j}, b_{i,j})$, the vector of color component values over a predefined neighborhood. This is a good representation, because if adjacent pixels behave similarly color wise, the pixel
value vectors are oriented in the same direction and the vector product is small. Notice, that the data-fidelity term is missing here. It can be omitted because for a given pixel, we only take the unmeasured color component into account and don’t change the measured data.

A different regularized optimization problem was introduced by Farsiou et al. in [12]. There a set of low-resolution images of the same scene is available and therefore demosaicing and super-resolution can be done together in one step. This is shown to be a better approach than trying to solve the two problems separately. If these problems are solved separately, the low-resolution images that are to be converted to one high-resolution image are demosaiced before super-resolution happens and therefore any artefacts introduced by demosaicing are newly considered a part of the scene that is to be reconstructed by super-resolution [27]. An example of formulating a joint demosaicing and super-resolution problem as defined in [12], reformulated for a single image, is:

$$O = SHF + N,$$

(1.6)

where $O$ again denotes the observed data, $F$ denotes the original scene, $H$ denotes the camera sensor blur, $S$ is the sampling and color filtering operator, and $N$ is additive noise. The solution of the inverse problem to (1.6) is found by minimizing a cost function consisting of four terms: a data-fidelity term, spatial luminance term, spatial chrominance, and intercolor dependencies term. The first one enforces the estimated data to be close to the measured raw output image. The second one favors sharp edges in the luminance component\(^2\). The third one, on the other hand, favors smoothness in the color planes, and the last term enforces edges to be at the same location in all color channels.

In the approach later described in Chapter 2.4 we will formulate the problem in a similar way as in Equation (1.6). We will concentrate on the Alternating Direction Method of Multipliers, which we will use to find an optimal solution for a regularized inverse problem.

---

\(^2\)In this method, luminance is defined, perhaps more correctly, as a linear combination of all three color components, specifically $L = 0.299R + 0.587G + 0.114B$
2. Joint Demosaicing, Denoising, Deconvolution, and Super-resolution

In this chapter, we formulate a joint solution for demosaicing, denoising, deconvolution, and super-resolution that will require solving a convex optimization problem. We will describe the Alternating Direction Method of Multipliers (ADMM), a method used to solve convex optimization problems. Finally, we will apply ADMM to our newly formulated convex optimization problem. We will later compare the results with other demosaicing methods in Chapter 3.

2.1 Formulation of the Problem

Consider the original scene \( F \) sampled by the camera and mosaiced by the CFA – these actions are represented by the sampling matrix \( S \). Operator \( H \) represents the camera sensor blur, and \( N \) is the additive noise. Then our raw output \( O \) from the camera was obtained from the original scene as follows:

\[
O = SHF + N.  \tag{2.1}
\]

Therefore to obtain the original scene, we must perform super-resolution, deblur, and denoise the observed data. As was already described in Chapter 1.3.2, we are looking for the solution to the inverse problem of Equation 2.1.

As was previously said, this is an ill-posed problem, and therefore we introduce regularization terms and solve the following problem [28]:

\[
\arg\min_{\hat{F}} \frac{1}{2} ||SH\hat{F} - O||^2_2 + \sum_i ||D\hat{F}_i||_1 + \mu(||D(\hat{F}_2 - \hat{F}_1)||_1 + ||D(\hat{F}_2 - \hat{F}_3)||_1),  \tag{2.2}
\]

where \( \hat{F}_i \) denotes the \( i \)th color channel of our estimate \( \hat{F} \), and \( D \) is the discrete version of the gradient operator. The first term is the data-fidelity term. We add two regularization terms. The second term is the so-called Total Variation (TV) regularization. This term is commonly used in image processing to enforce smooth images. The third term – the demosaicing regularization term – forces the edges in the color channels to be in the same location.

In the next section, we will provide a brief introduction into convex optimization, before describing ADMM itself in Section 2.3.

2.2 Convex Optimization

The exposition in this section is for the most part based on Boyd’s book on Convex Optimization [28]. A mathematical optimization problem can be written in the following standard form:

\[
\begin{align*}
\text{minimize} & \quad f(x), \\
\text{subject to} & \quad h_i(x) = 0, \quad i = 1, \ldots, m.
\end{align*}  \tag{2.3}
\]
The vector $x = (x_1, \ldots, x_n)$ is the optimization variable of the problem, the function $f : \mathbb{R}^n \to \mathbb{R}$ is called the objective function, and $h_i : \mathbb{R}^n \to \mathbb{R}$ are the constraint functions. A general optimization problem can also have inequality constraints. However, for our purposes, we will limit this text to only equality constrained problems. If $m = 0$, we call the problem an unconstrained optimization problem. Otherwise, it is a constrained optimization problem. Furthermore, we say that the optimization problem is convex if both the objective function and the constraint functions are convex.

Unconstrained optimization problems can be solved using many methods, of which most are based on first or second order derivative tests. Examples of first order derivative methods include the method of conjugate gradients and gradient descent. Methods based on second order derivatives of the objective function, which evaluate the matrix of second order derivatives – the Hessian – include for example Newton’s method.

In order to be able to find local and global maxima by testing the gradient, we generally try to turn a constrained optimization problem into an unconstrained one. This is the idea of the method of Lagrange multipliers, where we augment the objective function $f$ with terms representing the constraints. Thus, we define the Lagrangian $L(x, \nu)$ associated with the optimization problem in Equation 2.3, $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, as follows:

$$L(x, \nu) = f + \sum_{i=1}^{p} \nu_i h_i(x).$$

The solution of the original constrained problem is always a saddle point of $L(x, \nu)$ – where all the partial derivatives are equal to 0, but it is not a local extremum.

In the next section, we describe ADMM, which we will use to solve the problem in Equation 2.2.

### 2.3 Alternating Direction Method of Multipliers

In this section, we will provide a description of ADMM, a method used for solving convex optimization problems. The primary sources of information for this section is Boyd’s overview of ADMM [30], Jan Kotera’s doctoral thesis on image deblurring [31], and consultations with Filip Šroubek [28].

First, let us describe the idea behind the method informally. We will try to convert an optimization problem to an equivalent problem that is easier to minimize. In order to do that, we will first convert it, somewhat counter-intuitively, to a constrained problem. Then we “split” it into sub-problems using a technique called variable splitting. These sub-problems are then easier to minimize separately. Consider the following optimization problem:

$$\text{minimize } f(x) + g(Ax),$$

where $x$ is the optimization variable, $f : \mathbb{R}^n \to \mathbb{R} \cup \infty$, $g : \mathbb{R}^m \to \mathbb{R} \cup \infty$ are closed, proper and convex, and $A \in \mathbb{R}^{m \times n}$ is a full column rank linear
operator. We introduce a new variable $z$ and turn the problem in Equation 2.4 into a constrained problem as follows:

$$\begin{align*}
\text{minimize} & \quad f(x) + g(z), \\
\text{subject to} & \quad Ax - z = 0.
\end{align*}$$

(2.5)

We will now form the so-called augmented Lagrangian associated with the problem in Equation 2.5 as follows:

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax - z) + \frac{\rho}{2} ||Ax - z||_2^2.$$ 

Notice that for $\rho = 0$, $L_\rho(x, z, y)$ is the regular Lagrangian associated with the problem in Equation 2.5.

Each iteration of ADMM then consists of three steps:

$$\begin{align*}
x^{k+1} &= \arg \min_x L_\rho(x, z^k, y^k) \quad (2.6) \\
z^{k+1} &= \arg \min_z L_\rho(x^{k+1}, z, y^k) \quad (2.7) \\
y^{k+1} &= y^k + \rho(Ax^{k+1} - z) \quad (2.8)
\end{align*}$$

By repeatedly applying the steps in Equations 2.6-2.8, if specific conditions are satisfied, we converge to a solution of the optimization problem. In the next section, we show when convergence of ADMM is guaranteed.

### 2.3.1 Convergence

Below, we give the theorem proving convergence of the optimization method given in the previous section. It is replicated from [30], which also contains the proof.

**Theorem 1.** Consider an optimization problem, like the one in Equation 2.4. Let $f$ and $g$ be closed, proper, convex functions, and $f : \mathbb{R}^n \to \mathbb{R} \cup \infty$, $g : \mathbb{R}^m \to \mathbb{R} \cup \infty$, and $A \in \mathbb{R}^{m \times n}$ is a full column rank linear operator. Let there exist a saddle point $(x^*, z^*, y^*)$ such that:

$$L_0(x^*, z, y^*) \leq L_0(x, z, y^*) \text{ and, } L_0(x, z^*, y^*) \leq L_0(x, z, y^*), \forall x \in \mathbb{R}^n, \forall z \in \mathbb{R}^m.$$ 

Let $\alpha_k$ and $\beta_k$ be the upper bounds for the errors:

$$\begin{align*}
||x^k - \arg \min_x L_\rho(x, z^k, y^k)|| &\leq \alpha_k, \\
||z^k - \arg \min_z L_\rho(x^{k+1}, z, y^k)|| &\leq \beta_k.
\end{align*}$$

And let these bounds satisfy the following:

$$\sum_{k=1}^\infty \alpha_k \leq \infty, \sum_{k=1}^\infty \beta_k \leq \infty.$$ 

Then under these assumptions, the sequences $x^k, z^k$ and $y^k$ satisfy the following:

$$x^k \to x^*, z^k \to z^* \text{ and } y^k \to y^*.$$
2.4 Application of ADMM

As already formulated in Section 2.1, joint demosaicing, denoising, deconvolution and super-resolution can lead to the following convex optimization problem:

$$\arg\min_{\hat{F}} \frac{\gamma}{2} \|SH\hat{F} - O\|_2^2 + \sum_i \|D\hat{F}_i\|_1 + \mu(\|D(\hat{F}_2 - \hat{F}_1)\|_1 + \|D(\hat{F}_2 - \hat{F}_3)\|_1). \quad (2.9)$$

In this section, we will apply ADMM to the above problem. First, we will turn it into a constrained problem and introduce a new variable $Z$, as follows:

$\minimize \frac{\gamma}{2} \|SH\hat{F} - O\|_2^2 + \sum_i \|Z_i\|_1 + \mu(\|Z_2 - Z_1\|_1 + \|Z_2 - Z_3\|_1)$,

subject to $D\hat{F} - Z = 0$. \quad (2.10)

We now form the augmented Lagrangian associated with the problem in Equation (2.10) as follows:

$$L_\rho(\hat{F}, Z, \nu) = \frac{\gamma}{2} \|SH\hat{F} - O\|_2^2 + \sum_i \|Z_i\|_1 + \mu(\|Z_2 - Z_1\|_1 + \|Z_2 - Z_3\|_1) + \nu^T(D\hat{F} - Z) + \frac{\rho}{2} \|D\hat{F} - Z\|_2^2.$$  

We then convert this into a so-called scaled form, by completing the square on the last term.

$$L_\rho(\hat{F}, Z, \tilde{\nu}) = \frac{\gamma}{2} \|SH\hat{F} - O\|_2^2 + \sum_i \|Z_i\|_1 + \mu(\|Z_2 - Z_1\|_1 + \|Z_2 - Z_3\|_1) + \frac{\rho}{2} \|D\hat{F} - Z\|_2^2 + \tilde{\nu}^T(D\hat{F} - Z) + \frac{\rho}{2} \|D\hat{F} - Z\|_2^2,$$

where $\tilde{\nu} = \frac{1}{\rho} \nu$ is the newly scaled dual variable.

The point of converting the problem in Equation (2.9) to the one in Equation (2.10) is that we have successfully separated the smooth data-fidelity term from the non-continuously differentiable $l_1$ norm terms. We can now minimize both the sub-problems separately, which is much easier. The iterations for this problem as presented in Equation (2.9) will look as follows:

$$\hat{F}^{k+1} = \arg\min_{\hat{F}} \frac{\gamma}{2} \|SH\hat{F} - O\|_2^2 + \frac{\rho}{2} \|D\hat{F} - Z + \tilde{\nu}\|_2^2, \quad (2.11)$$

$$Z^{k+1} = \arg\min_{Z} \sum_i \|Z_i\|_1 + \mu(\|Z_2 - Z_1\|_1 + \|Z_2 - Z_3\|_1) + \frac{\rho}{2} \|D\hat{F} - Z + \tilde{\nu}\|_2^2, \quad (2.12)$$

$$\tilde{\nu}^{k+1} = \tilde{\nu}^k + D\hat{F}^{k+1} - Z^{k+1}. \quad (2.13)$$

The minimization step in Equation (2.11) after differentiating with respect to $\hat{F}$, comes down to solving the following equation:

$$(H^T S^T S H + \rho D^T D) \hat{F} = H^T S O + \rho D^T (Z - \tilde{\nu}),$$
which is done via the method of Conjugate Gradients. The problematic minimiza-
tion of the sum of $l_1$ norms in Equation 2.12 is done by eliminating the absolute
values and obtaining the derivative case-by-case. Finally, the dual udpate in
Equation 2.13 is simply a linear operation.

Under the assumption that the solution exists and since both the data-fidelity
term and the regularization terms are proper, convex and closed functions, and
since in each iteration we minimize the Lagrangian with respect to $\tilde{F}$ and $Z$, the
convergence follows from Theorem 1.
3. Experimental Results

In this chapter, we take images from the Kodak image dataset [6] as our original scenes. We sample the images according to the Bayer CFA and perform demosaicing on them. We compare the results obtained from 4 different methods:

1. Matlab’s native demosaicing method (MTLB),

2. Hirakawa’s adaptive homogeneity-directed demosaicing (AHD) [9], as described in Chapter 1.2.2

3. Joint solution for demosaicking, denoising and super-resolution via a neural network (TEN) [15],

4. Joint demosaicing, denoising, deconvolution and super-resolution via the ADMM (ADMM), as described in Chapter 2.

Results for 1) were obtained using Matlab’s demosaic function. Code for 2) was provided by the author, Keigo Hirakawa. Code for 3) is available on Github [15], and code for 4) was provided by Filip Šroubek.

In Section 3.1, we explain how the raw camera output is simulated. In Section 3.2, we compare the tested methods on all 24 images from the dataset by measuring the PSNR and SSIM metrics. In the last section, we compare some of the reconstructed images visually, by zooming in on a specified region of interest, which best shows the differences.

3.1 Simulation of the Raw Output

We need a reference ground truth that would represent the original scene, in order to make a visual comparison of the reconstructed image, as well as to apply reference quality metrics. We will represent our original scenes by the images from the Kodak dataset [6]. We simulate the sensor blur by averaging pixels in a 2x2 neighborhood and we mosaic the image according to the Bayer CFA.

3.2 Quantitative Comparison

In this section, we quantitatively measure the quality of our reconstructed images with reference to the ground truth. For this purpose, we will use peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM), metrics commonly used in image processing to measure the quality of a reconstructed image.

PSNR is commonly used to measure the quality of reconstructed data. It is given by [5]:

$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX^2}{MSE} \right),$$  \hspace{1cm} (3.1)

where $MAX^2$ denotes the maximum possible value of a pixel in the image, also called the dynamic range [5]. $MSE$ denotes the mean squared error between the
The value of MSE is as follows:

\[
MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (F_{i,j} - \hat{F}_{i,j})^2
\]  

(3.2)

The values of PSNR are in dB. For color images, where the pixel values are between 0 and 255, the PSNR values are traditionally between 20 and 50, where higher value signifies a better-quality result.

SSIM is a metric that was developed to capture quality differences more similarly to human perception [32]. The SSIM index takes values between 0 and 1, where 1 is achieved only when identical images are compared. Again, the higher the index, the better the quality. It is to be noted that SSIM is used for comparing grayscale images. The numbers we will show were therefore obtained by comparing only the green channels.

The results can be seen in Table 3.1 and Table 3.2, the best achieved value across all compared methods for a given original scene is in bold. We can see that the reconstructions obtained by ADMM achieve the second best results in PSNR, being outperformed only by TEN. In two instances, the ADMM reconstruction has the highest PSNR value. SSIM results are similar, but here ADMM outperformed all other tested methods in all but one instance. Second best SSIM values were obtained by the TEN reconstructions.

We can see that the difference in both PSNR and SSIM between the two best methods and the rest is quite significant. One possible interpretation of these results is that both TEN and ADMM aren’t just demosaicing methods. They perform demosaicing, denoising and super-resolution jointly, and in the case of ADMM also deconvolution. Furthermore, ADMM takes the exact sensor blur parameters as input, so it removes the blur efficiently, without estimation. This can be later seen in Section 3.3 where the ADMM results appear to be the sharpest. However, the ADMM reconstructions also suffer from quite a significant amount of color artefacts. However, the color artefacts have lesser impact on the SSIM value in this case, as the SSIM index is calculated for green channels only. The unfortunate reality is that neither PSNR, or SSIM, or any other reference metric, is perfect in matching visual comparison. This is one of the current issues in image processing [28].

### 3.3 Visual Comparison

In this section, we compare the four methods on four chosen images from the Kodak dataset [6] on a specified region of interest. Although the difference can be often subtle, we will try to point some of them out. The differences will be most visible when zooming in on the images, if viewing the electronic version of this thesis.

Results from MTLB and AHD are very similar. On sharper images, MTLB produces some color artefacts. Although barely visible, it can be seen in Figure 3.3c in the inner corner of the eye and in the area of reflection. AHD is very consistent and thanks to the median filtering step has almost no visible artefacts.

TEN rarely produces color artefacts, the results are very smooth. Some color distortion, however, can be seen in Figure 3.2e. Color artefacts are most visible...
<table>
<thead>
<tr>
<th></th>
<th>MTLB</th>
<th>AHD</th>
<th>TEN</th>
<th>ADMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24.9486</td>
<td>25.0505</td>
<td><strong>27.2934</strong></td>
<td>26.2435</td>
</tr>
<tr>
<td>2</td>
<td>31.0314</td>
<td>31.2053</td>
<td>31.8158</td>
<td><strong>32.3291</strong></td>
</tr>
<tr>
<td>3</td>
<td>32.2953</td>
<td>32.4598</td>
<td><strong>34.2284</strong></td>
<td>32.9789</td>
</tr>
<tr>
<td>4</td>
<td>31.0954</td>
<td>31.0349</td>
<td><strong>32.9059</strong></td>
<td>32.1935</td>
</tr>
<tr>
<td>5</td>
<td>24.9639</td>
<td>24.9817</td>
<td><strong>28.3664</strong></td>
<td>25.7662</td>
</tr>
<tr>
<td>6</td>
<td>26.1768</td>
<td>26.2478</td>
<td><strong>28.0159</strong></td>
<td>27.5912</td>
</tr>
<tr>
<td>7</td>
<td>29.7382</td>
<td>29.7795</td>
<td><strong>34.2956</strong></td>
<td>31.2467</td>
</tr>
<tr>
<td>8</td>
<td>21.9615</td>
<td>22.0430</td>
<td><strong>25.9251</strong></td>
<td>23.5982</td>
</tr>
<tr>
<td>9</td>
<td>29.7080</td>
<td>29.8380</td>
<td><strong>33.4337</strong></td>
<td>31.3058</td>
</tr>
<tr>
<td>10</td>
<td>29.4288</td>
<td>29.5000</td>
<td><strong>34.0628</strong></td>
<td>31.2597</td>
</tr>
<tr>
<td>11</td>
<td>27.7051</td>
<td>27.7611</td>
<td><strong>29.7981</strong></td>
<td>28.9134</td>
</tr>
<tr>
<td>12</td>
<td>30.5799</td>
<td>30.6839</td>
<td><strong>33.6891</strong></td>
<td>32.0196</td>
</tr>
<tr>
<td>13</td>
<td>22.9824</td>
<td>23.0168</td>
<td><strong>24.0349</strong></td>
<td>23.9611</td>
</tr>
<tr>
<td>14</td>
<td>27.1421</td>
<td>27.0950</td>
<td><strong>29.1332</strong></td>
<td>28.3848</td>
</tr>
<tr>
<td>15</td>
<td>29.0143</td>
<td>29.0366</td>
<td><strong>31.2529</strong></td>
<td>30.3437</td>
</tr>
<tr>
<td>16</td>
<td>29.7816</td>
<td>29.9000</td>
<td>30.6821</td>
<td><strong>31.1833</strong></td>
</tr>
<tr>
<td>17</td>
<td>29.7795</td>
<td>29.7986</td>
<td><strong>32.9907</strong></td>
<td>30.8972</td>
</tr>
<tr>
<td>18</td>
<td>26.3510</td>
<td>26.2713</td>
<td><strong>27.9901</strong></td>
<td>27.7171</td>
</tr>
<tr>
<td>19</td>
<td>26.5042</td>
<td>26.6285</td>
<td><strong>30.7083</strong></td>
<td>27.6555</td>
</tr>
<tr>
<td>20</td>
<td>28.1681</td>
<td>28.4189</td>
<td><strong>32.4471</strong></td>
<td>29.3961</td>
</tr>
<tr>
<td>22</td>
<td>28.7715</td>
<td>28.7645</td>
<td><strong>30.6665</strong></td>
<td>30.0596</td>
</tr>
<tr>
<td>23</td>
<td>31.9498</td>
<td>32.0716</td>
<td><strong>35.6215</strong></td>
<td>33.3227</td>
</tr>
<tr>
<td>24</td>
<td>25.5562</td>
<td>25.5183</td>
<td><strong>27.0595</strong></td>
<td>26.2715</td>
</tr>
</tbody>
</table>

Table 3.1: Table of PSNR values for reconstructed images.
<table>
<thead>
<tr>
<th></th>
<th>MTLB</th>
<th>AHD</th>
<th>TEN</th>
<th>ADMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.7959</td>
<td>0.7948</td>
<td>0.8371</td>
<td>0.9309</td>
</tr>
<tr>
<td>2</td>
<td>0.8741</td>
<td>0.8755</td>
<td>0.8769</td>
<td>0.9487</td>
</tr>
<tr>
<td>3</td>
<td>0.9269</td>
<td>0.9282</td>
<td>0.9434</td>
<td>0.9595</td>
</tr>
<tr>
<td>4</td>
<td>0.8885</td>
<td>0.8873</td>
<td>0.9051</td>
<td>0.9502</td>
</tr>
<tr>
<td>5</td>
<td>0.8434</td>
<td>0.8418</td>
<td>0.9085</td>
<td>0.9404</td>
</tr>
<tr>
<td>6</td>
<td>0.8142</td>
<td>0.8141</td>
<td>0.8476</td>
<td>0.9416</td>
</tr>
<tr>
<td>7</td>
<td>0.9248</td>
<td>0.9244</td>
<td>0.9631</td>
<td>0.9672</td>
</tr>
<tr>
<td>8</td>
<td>0.7832</td>
<td>0.7828</td>
<td>0.8729</td>
<td>0.9345</td>
</tr>
<tr>
<td>9</td>
<td>0.9014</td>
<td>0.9017</td>
<td>0.9290</td>
<td>0.9559</td>
</tr>
<tr>
<td>10</td>
<td>0.8920</td>
<td>0.8919</td>
<td>0.9424</td>
<td>0.9586</td>
</tr>
<tr>
<td>11</td>
<td>0.8421</td>
<td>0.8420</td>
<td>0.8729</td>
<td>0.9452</td>
</tr>
<tr>
<td>12</td>
<td>0.8957</td>
<td>0.8957</td>
<td>0.9103</td>
<td>0.9627</td>
</tr>
<tr>
<td>13</td>
<td>0.7470</td>
<td>0.7432</td>
<td>0.7783</td>
<td>0.9027</td>
</tr>
<tr>
<td>14</td>
<td>0.8367</td>
<td>0.8364</td>
<td>0.8729</td>
<td>0.9408</td>
</tr>
<tr>
<td>15</td>
<td>0.8924</td>
<td>0.8918</td>
<td>0.9093</td>
<td>0.9504</td>
</tr>
<tr>
<td>16</td>
<td>0.8534</td>
<td>0.8544</td>
<td>0.8677</td>
<td>0.9496</td>
</tr>
<tr>
<td>17</td>
<td>0.9017</td>
<td>0.9012</td>
<td>0.9397</td>
<td>0.9537</td>
</tr>
<tr>
<td>18</td>
<td>0.8347</td>
<td>0.8316</td>
<td>0.8714</td>
<td>0.9428</td>
</tr>
<tr>
<td>19</td>
<td>0.8468</td>
<td>0.8464</td>
<td>0.8863</td>
<td>0.9365</td>
</tr>
<tr>
<td>20</td>
<td>0.9049</td>
<td>0.9054</td>
<td>0.9275</td>
<td>0.9481</td>
</tr>
<tr>
<td>21</td>
<td>0.8619</td>
<td>0.8604</td>
<td>0.8920</td>
<td>0.9417</td>
</tr>
<tr>
<td>22</td>
<td>0.8509</td>
<td>0.8479</td>
<td>0.8677</td>
<td>0.9423</td>
</tr>
<tr>
<td>23</td>
<td>0.9414</td>
<td>0.9408</td>
<td>0.9532</td>
<td>0.9532</td>
</tr>
<tr>
<td>24</td>
<td>0.8396</td>
<td>0.8373</td>
<td>0.8910</td>
<td>0.9445</td>
</tr>
</tbody>
</table>

Table 3.2: Table of SSIM values for reconstructed images.
in the results of ADMM, unfortunately. For example in Figure 3.1f on the edges of the door knob, or in Figure 3.2f both on the upper and lower lash line. In Figure 3.3f, it is again the reflection area and the inner corner of the eye, as well as the white paint to the left of the eye. Lastly, in Figure 3.4f the zipper artefact is visible on the contour of the nose and mouth, as well as on the eyebrows and the golden leaves of the crown. In our opinion, however, ADMM reconstructions are visibly less blurry than MTLB and AHD. On the other hand, TEN tends to oversmooth the images to the point of removing texture from the scene. This can for example be observed in Figure 3.1e, where the shape of the door knob is distorted, or in Figure 3.2e, where the eye, and especially the iris, seem very unnatural and glassy.

3.4 Summary

The ADMM results achieved very good PSNR and SSIM values. In the case of SSIM, ADMM outperformed all compared methods. One might argue that comparing single solutions like MTLB and AHD to joint solutions like TEN and ADMM is not very fair. However, the aim of this thesis was to compare results obtained from ADMM with other existing demosaicing approaches, and these include single solutions as well. Furthermore, ADMM beat the joint demosaicing and denoising solution TEN in SSIM, as well as in visual comparison for some tested images. The advantage of ADMM over TEN is that the ADMM reconstructions preserve texture and are sharp.

The noticeable downfall of the ADMM reconstructions are the color artefacts that appear in some edge areas. The artefacts could perhaps be reduced by a different choice of a demosaicing regularization term, or even different implementation. This discussion, however, is outside the scope of this thesis.

We would like to conclude this chapter by saying that joint solutions represent a group of potentially very strong methods, that are capable of producing very high-quality reconstructions of any scene. It is therefore expected that future research in image processing will continue down this path.
Figure 3.1: Reconstructions of Image 2 from the Kodak dataset.
Figure 3.2: Reconstructions of Image 4 from the Kodak dataset.
Figure 3.3: Reconstructions of Image 15 from the Kodak dataset.
Figure 3.4: Reconstructions of Image 17 from the Kodak dataset.
Conclusion

In this thesis we focused mainly on one part of the image processing pipeline – demosaicing. We provided a review of existing demosaicing methods in the first chapter. In the second chapter, we formulated a joint model for demosaicing, denoising, deconvolution and super-resolution in the form of a convex optimization problem. We presented and described a specific method useful in solving such problems - Alternating Direction Method of Multipliers. In the final chapter we compared four different demosaicing methods, both quantitatively and visually. Our joint method achieved great results in PSNR and SSIM reference metrics. Unfortunately, although sharp, the ADMM-reconstructed images have visible color artefacts. Despite these visible artefacts, the presented joint solution highlights the potential of joint solutions in image processing. Such solutions are popular, because they are more compact and prevent error trailing, which occurs when any artefacts get augmented by the successive steps in the image processing pipeline.
Bibliography


[28] Filip Šroubek. Personal Correspondence.


<table>
<thead>
<tr>
<th>List of Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Bayer Color Filter Array</td>
<td>2</td>
</tr>
<tr>
<td>2. Crisp original image in (a) and visible zippering effect on bilinearly</td>
<td>4</td>
</tr>
<tr>
<td>interpolated image in (b).</td>
<td></td>
</tr>
<tr>
<td>3. Original scene in (a) and visible false color artefact on reconstructed</td>
<td>4</td>
</tr>
<tr>
<td>image in (b).</td>
<td></td>
</tr>
<tr>
<td>1.1 Original scene (a), mosaiced raw output from a single-sensor digital</td>
<td>6</td>
</tr>
<tr>
<td>camera (b), and zoomed-in raw output.</td>
<td></td>
</tr>
<tr>
<td>1.2 Subsampled color channels obtained from Bayer’s CFA.</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Cutout from Bayer’s CFA.</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Original scene (a), corrupted reconstructed image (b).</td>
<td>9</td>
</tr>
<tr>
<td>B - G channels of the original scene (c), and of the reconstructed image</td>
<td></td>
</tr>
<tr>
<td>(d).</td>
<td></td>
</tr>
<tr>
<td>1.5 Corrupted reconstructed image in (a), and image reconstructed</td>
<td>10</td>
</tr>
<tr>
<td>from median filtered R - G and B - G channels in (b).</td>
<td></td>
</tr>
<tr>
<td>3.1 Reconstructions of Image 2 from the Kodak dataset.</td>
<td>26</td>
</tr>
<tr>
<td>3.2 Reconstructions of Image 4 from the Kodak dataset.</td>
<td>27</td>
</tr>
<tr>
<td>3.3 Reconstructions of Image 15 from the Kodak dataset.</td>
<td>28</td>
</tr>
<tr>
<td>3.4 Reconstructions of Image 17 from the Kodak dataset.</td>
<td>29</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Table of PSNR values for reconstructed images. . . . . . . . . . . 23
3.2 Table of SSIM values for reconstructed images. . . . . . . . . . . 24